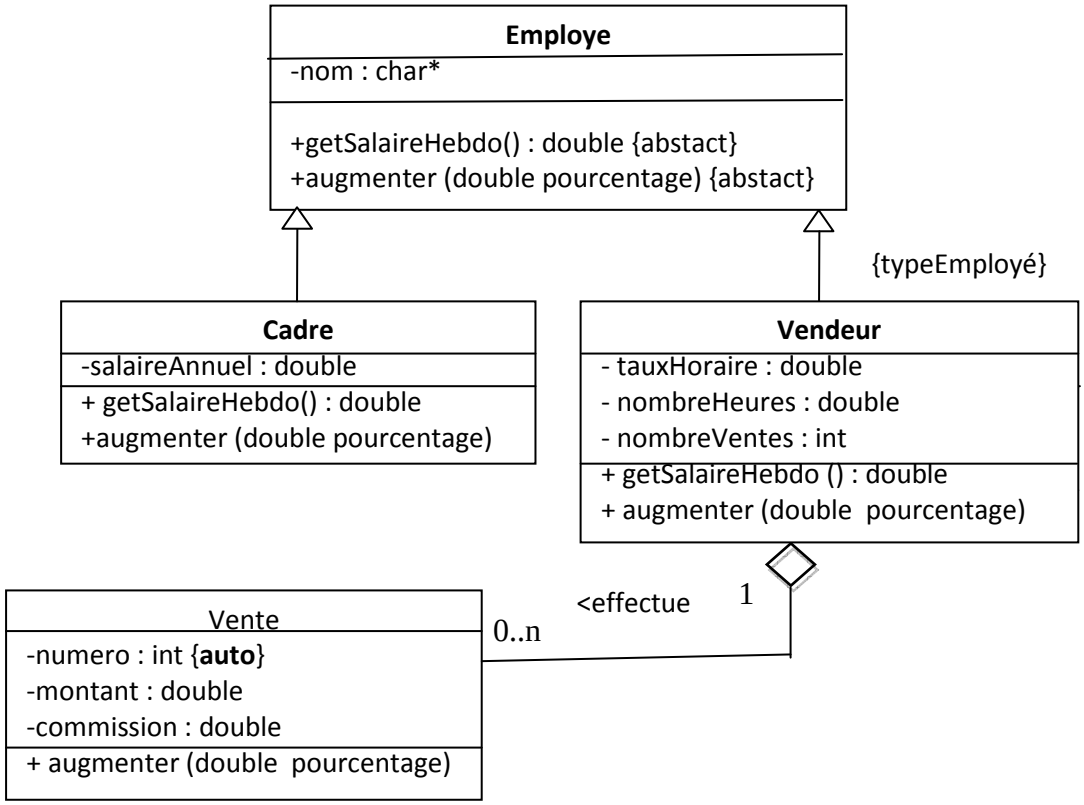
TRAVAUX PRATIQUES 1.

Évaluation : **20 points**

Date de distribution : 29/10/2014 **Date de remise : 19/11/2014**

Remettre : le rapport et l'application bien commentée en C++ avec le jeu de tests.

Soit une application **GestionEmployes** qui gère des employés.
Le diagramme des classes (notation UML) de l'application est le suivant:



Précisions :

- Dans la classe **Vente** :
  - l'attribut **numero** spécifie le numéro de la vente ; il est attribué automatiquement au moment de la création d'un objet Vente ;
  - l'attribut **montant** spécifie le montant d'une vente ;
  - l'attribut **commission** spécifie la commission (0.02, 0.05, 0.1, 0.15, 0.2,…) d'une vente. **Attention !** La commission peut varier d'une vente à une autre.
- Dans la classe **Vendeur,** il faut avoir les attributs suivants **:**
  - les attributs **tauxHoraire, nombreHeures, nombreVentes** ;
  - l'attribut – pointeur vers le tableau de Ventes **ventes** qui réalise l'agrégation et est déclaré comme suit : **Vente * ventes.**

  Attention ! Le nombre de ventes de chaque vendeur est différent !
- La méthode **getSalaireHebdo()** retourne :
  - Pour un cadre le salaire hebdomadaire calculé comme suit : salaireAnnuel/52 ;
  - Pour un vendeur le salaire hebdomadaire calculé comme suit : tauxHoraire*nombreHeures + total de commissions de toutes ses ventes.

- La méthode **augmenter(double pourcentage)** augmente :
  - Pour un cadre, le salaire annuel de pourcentage passé en paramètre ;
  - Pour un vendeur, le taux horaire de pourcentage passé en paramètre ;
  - Pour une vente, la commission de pourcentage passé en paramètre : la nouvelle commission est égale à l'ancienne plus le de pourcentage passé en paramètre .
- Chaque classe doit contenir des accesseurs (**getters**) et mutateurs (**settters**) nécessaires
- Chaque classe doit contenir les constructeurs et, si nécessaire, le constructeur de recopie, le destructeur et l'opérateur d'affectation surchargé (forme canonique de Caplien).
- Chaque classe **doit surcharger l'opérateur <<** pour l'affichage de ces objets.

Travail à faire :

- Créer une application orientée objet en C++ en utilisant les structures de données les plus appropriées (les classes avec l'héritage et polymorphisme et l'agrégation).
- Dans la fonction **main()** qui sera le véritable programme principal de l'application créer et tester:
  - Des objets sur la pile et sur le tas de la classe Vente avec toutes ses méthodes et l'opérateur << ;
  - Des objets sur la pile et sur le tas de la classe Cadre avec toutes ses méthodes et l'opérateur << ;
  - Des objets sur la pile et sur le tas de la classe Vendeur avec toutes ses méthodes et l'opérateur << ;
  - Le tableau des pointeurs vers des objets des classes Cadre et Vendeur en utilisant le polymorphisme.

Remarques :

- Le diagramme ne précise pas quelles classes doivent être en forme canonique de Coplien. À vous de le déterminer.
- À vous de déterminer aussi quelles méthodes doivent être virtuelles.
- N'oubliez pas de vérifier la validité des paramètres et d'utiliser les sorties formatées au besoin**.**

Consignes :

- Toutes les classes qui apparaissent dans le diagramme, ainsi que leurs attributs et leurs méthodes doivent être implémentés, tester et, probablement, utilisés par l'application.
- Ne changez pas les noms des classes, des attributs et des méthodes et ne modifiez pas les prototypes des méthodes (leurs signatures et leurs types de valeurs de retour).
- Séparez l'implémentation des classes en fichiers d'en-têtes (.h) et de code (.cpp) à l'exception des classes qui n'ont pas d'implémentation ou si leurs méthodes sont implémentées **inline**.

Bonus (2 points).

- (**1 point**) Ajouter les dates d'embauche des employés et les dates de vente dans les classes appropriées.
- (**1 point**) Ajouter une classe **CadreVendeur** qui hérite de classes **Cadre** et **Vendeur**. La méthode **getSalaireHebdo()** de la classe **CadreVendeur** retourne le salaire hebdomadaire calculé comme suit : salaireAnnuel/52 + total de commissions de toutes ses ventes. La méthode **augmenter(double pourcentage)** reste le même que dans la classe **Carde** (augmente le salaire annuel de pourcentage passé en paramètre).